DotBot in the Warehouse: A Swarm Robotics Platform for Intralogistics Research

Corentin Rasda
corentin.rasda@inria.fr
Inria, France

Trifun Savić
trifun.savic@ucg.ac.me
University of Montenegro
Montenegro

Geovane Fedrecheski
geovane.fedrecheski@inria.fr
Inria, France

Jean-Paul Fabre
jean-paul.fabre@laposte.fr
La Poste, France

Mališa Vučinić
malisa.vucinic@inria.fr
Inria, France

Thomas Watteyne
thomas.watteyne@inria.fr
Inria, France

Abstract

Swarm robotics promises warehouse-scale coordination using many low-cost units. However, existing robotic platforms are often limited by efficiency, cost and scalability. The low-cost DotBot platform enables intralogistics experiments with hundreds of robots, bridging the simulation-reality gap with minimal infrastructure and fast iteration cycles. We use DotBots to run Priority Inheritance with Backtracking (PIBT), a state-of-the-art multi-agent path-finding planner. We measure its *sim-to-real gap*, the difference in coordination performance between an abstract-grid simulation and the same planner driving real DotBots, considering an occupancy axis $\rho = N/\text{cells}$. Across 64 runs on a 5×5 grid, simulation and hardware agree at low density, but diverge as the grid fills up: the real swarm tracks the simulated reference up to an occupancy of $\rho = 0.24$ (six robots), beyond which run-to-run variance rises. Kinematic heterogeneity and network-induced timeouts widen the gap. On this 5×5 testbed, $\rho = 0.24$ marks the highest density at which the real swarm still matches simulation; whether this threshold holds at larger scales, robot counts, or arena shapes remains to be tested.

CCS Concepts

• **Computer systems organization → Robotic control**.

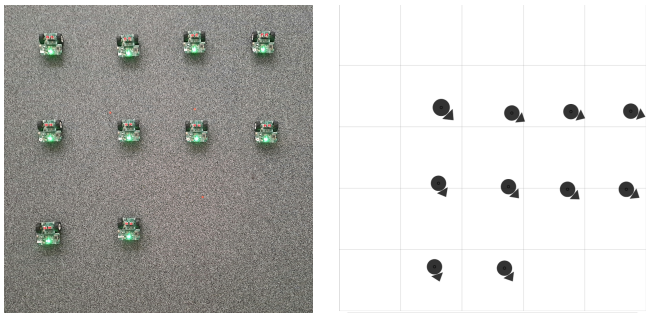
Keywords

Swarm robotics, DotBot, Intralogistics, PIBT.

1 Introduction

Swarm robotics trades a few heavily instrumented robots for many small, cheap, cooperative units, so throughput scales with swarm size rather than per-unit capability. This fits intralogistics, where tasks grow with warehouse size: a swarm of low-cost units absorbs growing demand without per-unit upgrades. We target semi-automated warehouses and parcel-sorting floors of the *Kiva* type, where fleets of mobile robots organize goods between storage shelves and sorting stations [1, 2]. Postal operators such as La Poste, whose sorting centers absorb large and bursty volumes, have a concrete demand for exactly these types of cheaper, smaller robots.

Although PIBT has won recent MAPF challenges and sees widespread industrial warehouse adoption, formal physical demonstrations remain scarce. Okumura *et al.* [3] validated it on a small centralized Toio fleet, but standard platforms (e.g., TurtleBot3 [1] or E-puck 2[2]) remain too costly or network-limited for dense swarm deployments.



(a) Real testbed **(b) Live map view**

Figure 1: DotBot testbed: a 2×2 m^2 grid arena with ten DotBot robots. Left: top-down picture of the physical testbed. Right: corresponding live map view showing the ten active DotBots.

The DotBot fills both gaps. At ~40 EUR per unit [4], it is significantly cheaper. Its Mari link layer (TSCH over BLE) provides a collision-free schedule scaling to dense deployments [5], connecting over 700 nodes (100 per gateway). The robot localizes itself from a Lighthouse v2 base station and reports it over Mari [4]. Coupled with the open-source PyDotBot SDK[3], it forms a ready-to-use, fast-iterating platform for intralogistics research. Beyond its affordability, the platform's major technical advantage is its REST API, which allows seamless switching between a software simulator and real hardware without rewriting the planner.

Our goal is to determine if the affordable DotBot platform can physically execute a state-of-the-art Multi-Agent Path Finding (MAPF) algorithm at high densities without prohibitive errors. To drive this baseline, we coordinate the swarm using Priority Inheritance with Backtracking (PIBT) [3] for multi-agent path finding (MAPF).

Our demonstration features DotBots executing PIBT (Fig. 1) at varying densities. Section 2 details the PIBT algorithm. Section 3 evaluates the platform's ease of setup, research accessibility, and physical scalability. We quantify this scaling via the *sim-to-real gap*: the performance disparity between an abstract-grid simulation and real hardware, measured along a shared occupancy axis $\rho = N/\text{cells}$.

[1] https://www.turtlebot.com

[2] https://www.gctronic.com/e-puck2.php

[3] https://github.com/DotBots/PyDotBot

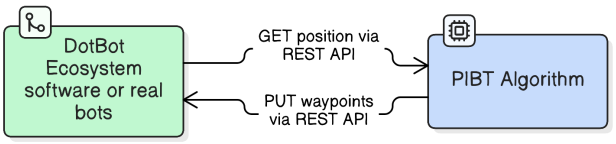


Figure 2: PIBT needs two-way access to the DotBot ecosystem: it reads each robot's position (`GET`) and sends the next waypoint (`PUT`). Both calls go through the DotBot Controller REST API: a software simulator and real DotBots are interchangeable backends for the unmodified planning code.

2 PIBT Algorithm

PIBT routes N agents from their start cells to N distinct goal cells on a shared graph without vertex or swap collisions [6]. Fig. 2 shows PIBT's two-way dependency on the swarm: it reads each robot's position and, in return, sends it a waypoint. We adopt it because it extends naturally from one-shot path finding to lifelong pickup-and-delivery, where robots are continuously assigned new tasks on the fly. This continuous formulation is the closest abstraction to a working automated warehouse.

PIBT solves MAPF iteratively in unit-length time windows [3]. Agents move synchronously based on dynamic priorities. Blocked higher-priority agents lend their priority to blockers, forcing them to yield or backtrack. On biconnected graphs (like our grid), this scheme guarantees that all agents eventually reach their goals, barring rare deadlocks.

3 Results

The platform is controlled through the `DotBot SDK`, which includes a command-line interface (CLI), a simulator, and a control plane exposing a REST API. The user can select whether the REST API controls the real swarm or the simulator. We implement a PIBT module that interacts with the control plane, as shown in Fig. 2.

We use an identical PIBT code to drive two setups on the shared occupancy axis $\rho = N/\text{cells}$: *PIBT simulation*, an abstract-grid run that fixes the ideal reference, and *DotBots testing*, the same planner driving real robots. The physical arena is $2{\times}2$ m^2; DotBots testing uses a 5×5 grid (25 cells, 40 cm side). The swarm is driven by one synchronous PIBT step at a time through a barrier: each robot is sent to its next cell, and the loop waits until all have arrived (or an 8 s per-step timeout elapses) before re-reading positions and planning the next step. Goals are drawn uniformly at random on an obstacle-free grid—a deliberate worst-case for pathfinding. While simulations confirm PIBT successfully handles obstacles provided the graph remains biconnected, this open layout isolates the impact of physical density. The logs hold 7 configurations ($N = 4$–10) over **64 runs** total.

Setup. We use 10 DotBots, a laptop, one Mari gateway, and a Lighthouse v2 base station. A complete ten-robot swarm testbed costs less than a *single* platform surveyed in Section 1.

Experimentation. Our headline metric is the *makespan*, the number of PIBT steps to complete a scenario, reported with its standard deviation. We also log the *success rate*: the fraction of DotBots that reach their goal (arrived/N). In PIBT simulation, the reference remains stable: the success rate eases from 99.2% ($N = 4$, $\rho = 0.16$) to 93.3% ($N = 10$, $\rho = 0.40$) the residual failures being rare deadlocks under randomly drawn goals, not step-budget timeouts while the makespan grows only mildly, from 5.5 to 6.9 steps (std 1.3–1.8). DotBots testing tracks this reference closely up to $\rho = 0.24$ ($N = 6$): makespan 5.0–5.6 steps at low variance (std 0.8–1.1) and success at or above 95%. The divergence begins at $\rho = 0.28$ ($N = 7$): the mean makespan still tracks the reference (5.8 steps) but its spread jumps (std 2.7) and success drops to 90%. Beyond it ($N = 8$–10, $\rho \geq 0.32$) predictability collapses : the makespan climbs from 7.0 to 9.3 steps (std 5.2–6.6) and success falls to 87%, 78%, then 69%. Two physical, non-algorithmic effects compound with density: (i) *kinematic heterogeneity* — wear and battery spread make some robots slower, so a late robot collides with faster neighbors; and (ii) *timeout cascades* — unacknowledged packet drops cause uncommanded robots to fall behind the barrier, breaking local coherence.

Idealized simulations inherently ignore these real-world intralogistics problems, physical testbeds like the DotBot are strictly necessary to expose them.

4 Conclusion

We demonstrated that DotBot provides a low-cost, flexible architecture for physical intralogistics and swarm robotics research. By exposing a common REST API, the system allows researchers to transparently swap between a software simulation backend and real physical robots without altering the core algorithmic implementation. As proven by our deployment of the PIBT algorithm, this plug-and-play capability significantly simplifies the transition from theory to real-world validation, thereby supporting the integration of various pathfinding algorithms.

Our evaluation successfully captured the physical realities such as kinematic heterogeneity and communication drops that cause real-world performance to diverge from idealized simulations at high densities. Future work will directly address this limitation by scaling the DotBot testbed to a significantly larger swarm operating over extended timeframes. Furthermore, leveraging the unified API, we plan to implement and benchmark more complex, continuous task-allocation algorithms, such as Lifelong MAPF and Multi-Agent Pickup and Delivery (MAPD) [3], moving closer to the dynamic requirements of fully automated sorting centers.

References

[1] J. Wen, L. He, and F. Zhu, "Swarm robotics control and communications: Imminent challenges for next generation smart logistics," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 102–107, 2018.

[2] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.

[3] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority Inheritance with Backtracking for Iterative Multi-Agent Path Finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022.

[4] S. Alvarado-Marin, M. Balbi, A. Abadie, T. Savić, F. Maksimovic, and T. Watteyne, "DotBot, a cm-Scale, Easy-to-Use Micro-Robot for Swarm Research," 2024.

[5] G. Fedrecheski, Y. Gao, A. Abadie, S. Alvarado-Marin, M. Vučinić, F. Maksimovic, and T. Watteyne, "Demo: Mari allows connecting large scale robot swarms using TSCH over BLE and multiple independent gateways," in *International Conference on Embedded Wireless Systems and Networks (EWSN)*, 2025.

[6] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the international symposium on combinatorial search*, vol. 10, no. 1, 2019, pp. 151–158.